J.D. Lohn, J.A. Reggia, "Exploring the Design Space of Artificial Self-Replicating Structures," Evolution of Engineering and Information Systems and Their Applications, L.C. Jain (Ed), CRC Press, 2000, pp. 67-103.

EXPLORING THE DESIGN SPACE OF ARTIFICIAL SELF-REPLICATING STRUCTURES

Jason D. Lohn
Caelum Research Corporation
NASA Ames Research Center
MS 269-1, Moffett Field, CA 94035-1000 USA
jlohn@ptolemy.arc.nasa.gov

James A. Reggia
Computer Science Department and
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742 USA
reggia@cs.umd.edu

In this chapter we demonstrate how a genetic algorithm using a multi-objective fitness function can be used to automatically design self-replicating structures in cellular automata models. Past models of self-replicating structures have been manually designed, a difficult and time-consuming process. The self-replicating structures designed using our techniques compare favorably in terms of simplicity with those created manually in the past, but differ in interesting ways. These results suggest that further exploration in the space of possible self-replicating structures will yield additional new structures. Furthermore, the research described here sheds light on the process of creating self-replicating structures, which could potentially lead to future studies on the discovery of novel self-replicating molecules and self-replicating assemblers in nanotechnology.

1 Introduction

Self-replicating systems have the ability to produce copies of themselves. Biological organisms are the most familiar examples of such systems, and until the late 1940s, the only instances formally researched. Mathematicians and scientists began studying artificial self-replicating systems when it became desirable to gain a deeper understanding of complex systems and the fundamental information-processing principles involved in self-replication [46], [47]. The initial models consisted of abstract logical machines, or automata, embedded in cellular spaces [2], [8], [14], [19], [38]. Other computational models, such as self-replicating computer programs have also been investigated [17], [37], as well as mechanical and biochemical models [15], [32], [33]. The field of artificial life, which studies life-like behaviors (such as self-replication) from a computational perspective, grew largely out of work based on self-replicating cellular automata structures [20]. The automatic discovery of such systems could be useful in areas such as nanotechnology [9], programming massively parallel computers [36], and computer viruses [16].

Over the decades since von Neumann first demonstrated cellular automata structures that can self-replicate [47], theoretical and modeling studies have led to progressively simpler and smaller structures [8], [5], [19], [38]. They have produced structures that do problem solving while replicating [7], [34], [43], as well as demonstrated that self-replicating structures can emerge from a “sea” of non-replicating components [6]. However, all such past models have been manually designed, a process that is very difficult and time-consuming, and is prone to subjective biases of the implementor.

This research [26] presents the use of genetic algorithms [10], [13] to discover automata rules that govern emergent self-replicating processes. Identification of effective performance measures (fitness functions) for self-replicating structures is the key challenge in this problem. A genetic algorithm using multiobjective fitness criteria was applied to automate rule discovery. The experimental results show that statistically significant quantities of discovered structures were found, showing for the first time that genetic algorithms can be used to successfully automate the search for self-replicating structures.

The specific examples of self-replicating structures presented here provide evidence that our techniques are effective. While these self-replicating structures are specialized, the techniques by which they were evolved are not, being quite general and applicable to other structures in cellular automata. Evidence of this generality is presented by using the same fundamental technique for different size component structures. The main factor currently limiting the complexity and size of evolved structures is computer time.

The size of the search spaces for CA models (the set of all possible CA rule tables) can be incredibly large. Genetic algorithms are a well-known strategy for searching such extremely large search spaces. In addition to its size, the search space fitness landscape is not well understood. While there are detailed reports examining small (two state) cellular automata [48], little has been reported which attempts to understand the larger search spaces of models having more than two states. Such search spaces are very unlikely to be smooth and unimodal, which would suggest they cannot effectively be searched using gradient-ascent algorithms.

We provide a crossover technique that partitions the rule table genome into segments, one segment per cell state. This has the effect of evolving "state behaviors" independently, and proved to be more effective than any of the standard crossover techniques that were tried. This crossover technique may prove useful in general for genetic algorithm practitioners when evolving cellular automata rule sets. As researchers continue to evolve successively larger cellular automata models, having effective genetic operators can greatly speed up the search.

In this chapter, the goal of automatically finding self-replicating structures is not directly concerned with finding the optimal self-replicating structure, the definition of which would be subjective. Given that less than 30 self-replicating structures have been reported in the literature, finding a diverse set of such structures is of greater importance and more interesting. Our results show that novel self-replication processes were uncovered by the genetic algorithm. For example, some of our structures both rotate and move during self-replication, and some leave around unused components (debris) which promote the formation of new structures. Such behaviors, which have not been used or considered in past manually-designed self-replicating structures, are especially interesting, suggesting that evolutionary computation can discover novel design concepts of general value.

A new paradigm for cellular space models with certain rotational symmetries is introduced which significantly reduces rule table size without adversely affecting the key properties of the model. Called orientation-insensitive input, this technique reduces the search space size, thus facilitating the search process. Experimental results using genetic algorithms are presented which verify this.

The remainder of this chapter is organized as follows. Cellular automata and self-replicating structures are introduced and described in Section 2. Section 3 presents the genetic algorithm that was applied, and in Section 4 our experimental results are analyzed. We discuss our conclusions and future work in Sections 5 and 6.

2 Cellular Automata and Self-replicating Structures

2.1 Cellular Automata

Cellular automata (CA) are a class of spatially-distributed dynamical system models in which many simple components interact to produce potentially complex patterns of behavior[8], [48]. In a cellular automata model, time is discrete, and space is divided into an N-dimensional lattice of cells, each cell representing a finite state machine or automaton. All cells change state simultaneously with each using the same function δ or rule table to determine its next state as a function of its current state and the state of neighboring cells. This set of adjacent cells is called a *neighborhood*, the size of which, n, is commonly five or nine cells in 2-D models (see Figure 1). By convention, the center cell is included in its own neighborhood. Each cell can be in one of k possible states, one of which is designated the quiescent or inactive state. When a quiescent cell has an entirely quiescent neighborhood, a widely accepted convention is that it will remain quiescent at the next time step.

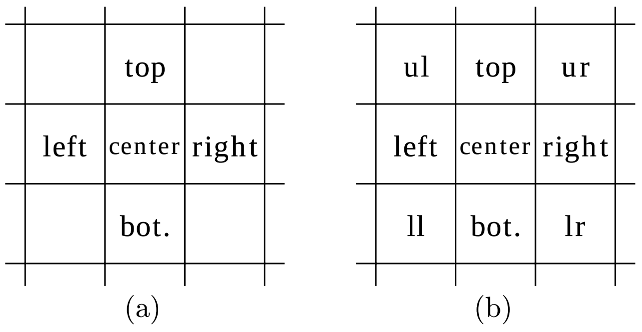


Figure 1: Common neighborhood templates in 2-D CAs: (a) 5-cell von Neumann neighborhood; (b) 9-cell Moore neighborhood

The CA rule table is a list of transition rules that specify the next state for every possible neighborhood combination. In a 2-D, 5-neighbor model the individual transition rules would be of the form CTRBL $\rightarrow$ C$'$, where CTRBL specifies the states of the Center, Top, Right, Bottom, and Left positions of the neighborhood's present state, and C$'$ represents the next state of the center cell.

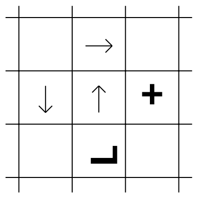
The underlying space of CA models is typically defined as being isotropic, meaning that the absolute directions of north, south, east, and west are indistinguishable. However, the rotational symmetry of cell states is frequently varied. Strong rotational symmetry implies that all cell states are unoriented, meaning that each neighbor to a cell has no distinguishable orientation. Weak rotational symmetry implies that at least one cell state[1] is directionally oriented, mean-

[1]The quiescent state is always a strongly rotation symmetric cell state and is generally included in CA models with weak rotational symmetry.

ing that the cell designates specific neighbors as being its top, right, bottom, and left neighbors. For example, the cell state designated $\uparrow$ in von Neumann's work is weakly-symmetric and thus permutes to different cell states $\rightarrow$, $\downarrow$, and $\leftarrow$ under successive 90° rotations. It represents one oriented *component* that can exist in four orientations. In CAs that contain both weak and strong rotationally symmetric states, it is common to represent the "strong" states using symbols that appear rotationally symmetric (e.g., $\circ$, $+$, $\times$), and the "weak" states (components) using symbols that are not rotationally symmetric (e.g., $\uparrow$, A, L).

In cellular automata models with weak rotational symmetry, an automaton is sensitive to the orientation of states of its neighboring cells, and uses this input to make a state transition. We call this method of cell input *orientation sensitive input.* We introduce an alternative method in which an automaton receives only information about its neighboring cell's component type, and not the component's orientation. Such automata are called *orientation insensitive.* Component types are represented by underlining cell state symbols. For example, the symbol $\underline{\mathtt{L}}$ represents the component type for the four oriented cell-states {L, ⌐, ˥, ⅃}, with each of the four cell-states being functionally identical. Figure 2 shows an example of a cell's input patterns under both types of input sensitivity. In the orientation-sensitive case, the center cell $\uparrow$ senses an ⅃ cell state below it. Under orientation-insensitive input, however, the center cell $\uparrow$ senses only the component type $\underline{\mathtt{L}}$. Orientation insensitivity is different from strong rotational symmetry since the positions of top, right, bottom, left, are not explicitly distinguished for cell states having strong rotational symmetry.

By decreasing the amount of input information each automaton receives using orientation insensitivity, the rule table is significantly reduced in size. Using smaller rule tables has advantages such as a decreased computational load and decreased search space size. In the context of manipulating large rule tables, and searching vast rule table search spaces, orientation insensitive input allows more effective experimentation while benefiting from using the same underlying CA space as standard CA models.



Input pattern TRBL under orientation-sensitive input:
$\rightarrow + ⌙ \downarrow$

Input pattern TRBL under orientation-insensitive input:
$\underline{\uparrow} + \underline{\text{L}}\ \underline{\uparrow}$

Figure 2: Example illustrating the effects of differing input sensitivity. The set of cell states is $\{\bullet, +, \text{L}, ⌐, ⅂, ⌙, \uparrow, \rightarrow, \downarrow, \leftarrow\}$, and the set of component types is $\{\underline{\text{L}}, \underline{\uparrow}\}$.

The amount of rule table reduction under orientation insensitive input is calculated by comparing the expressions for rule table sizes under both methods of input. First assume there is only one strongly rotation symmetric state. This is reasonable since it is common for weakly rotation symmetric models to have only one strongly symmetric state which represents the quiescent cell state. Letting $|\delta|$ represent rule table size, the ratio for the rule table sizes is $\frac{|\delta|_{osi}}{|\delta|_{oii}}$, where *osi* denotes orientation sensitive input and *oii* denotes orientation insensitive input. This ratio is derived in [25] and can be expressed as

$$\frac{{}_{(\beta c+1)}\mathrm{CP}_{n-1} + c(\beta c+1)^{n-1}}{{}_{(c+1)}\mathrm{CP}_{n-1} + c(c+1)^{n-1}} \tag{1}$$

where β denotes the number of coordinate systems rotations permitted, c is the number of component types, and ${}_k\mathrm{CP}_{n-1}$ denotes the circular permutation function [12] used to count distinct neighborhood patterns. This ratio converges to a constant as the number of components c is increased:

$$\lim_{c\to\infty} \frac{{}_{(\beta c+1)}\mathrm{CP}_{n-1} + c(\beta c+1)^{n-1}}{{}_{(c+1)}\mathrm{CP}_{n-1} + c(c+1)^{n-1}} = \beta^{n-1} \tag{2}$$

Thus as we increase the number of components, models using orientation insensitive input have rule tables that are approximately

β^{n-1} smaller than models using orientation sensitive input. For a typical coordinate system with $\beta = 4$, and using the von Neumann and Moore neighborhoods, it is seen that the $|\delta|$ ratios are 256 and 65536, respectively, as the number of components increases. This multiplicative increase translates into orders of magnitude increases in search space sizes:

$$\begin{aligned} |D_n^k|_{osi} &= k^{(|\delta|_{osi})} \\ &\simeq k^{\beta^{n-1}(|\delta|_{oii})} \\ &\simeq (|D_n^k|_{oii})^{\beta^{n-1}} \end{aligned} \tag{3}$$

where $|D_n^k|$ denotes the size of the set of all possible rule tables for a CA with k states and n neighbors. From Eq. (3) it is seen that by using orientation insensitive input, the search space is decreased by approximately β^{n-1} orders of magnitude. As an example, the models used in this work have $\beta = 4$ and $n = 5$, giving a difference of 256 orders of magnitude.

2.2 Self-replicating Structures

In CA models, one state is designated the quiescent state, and the remaining states are considered *active*. A self-replicating structure is represented as a contiguous configuration of active cells that goes through a sequence of steps to construct a duplicate of itself. The replica can be displaced and potentially rotated relative to the original at a later time t'. An example two-dimensional CA (from [38]) illustrating this is shown in Figure 3. This figure shows structure `UL06W8V`, so named because it is an unsheathed loop (UL), is comprised of six components, uses weak rotational symmetry (W), is embedded in a model in which each cell may be in one of eight states, and uses the von Neumann neighborhood (V). Twenty CA transition rules are used during its self-replication process. The structure that undergoes self-replication is seen at $t = 0$ in Figure 3. At $t = 8$ the first replicant can be seen, on the right, detached from the original structure. Then these two structures each begin a process of self-replication until, several time steps later, a diamond-shaped colony has formed.

```
......   ......   ......   ......   ......   ......   ......   ..O...
......   ......   ......   ......   ......   ...<..   ..#<..   ..O...
.OO...   .O<...   .vL...   .LO.O.   .OO^O<   .O<.vL   .vL.LO   .LO.OO
.L>OO.   .OL>O.   .OOL>.   .>OOL^   .L>OOL   .OLvOO   .OO.>O   .>O.L^
......   ......   ......   ......   ......   ......   ...>..   ...>#.
......   ......   ......   ......   ......   ......   ......   ......
 t = 0    t = 1    t = 2    t = 3    t = 4    t = 5    t = 6    t = 7

                  ..O...   ..O...   ..O...
                  ..O...   ..O...   ..^...
                  .OO.O<   .O^.vL   .vL.LO
                  .L^.OL   .OL.OO   .OO.vO
                  ....O.   ....O.   ....O.
                  ....O.   ....O.   ....O.
                   t = 8    t = 9   t = 10
```

Figure 3: The first 10 time steps of structure UL06W8V [38]. Signals `L>` circulate counterclockwise around the loop starting at t=0. By time t=10 a duplicate structure has appeared on the right (rotated), while the arm of the original structure has moved upwards.

Cellular space models of self-replicating systems have progressed from complex models to less-complex models. This trend is apparent in Figure 4, where complexity is plotted against time for cellular automata models. There are certainly other measures of complexity one could chose, but we have defined it to be the product of rule table size and structure size, plotted logarithmically. As can be seen, models designed only for self-replication are lowest in complexity, with the least complex of the others having three orders of magnitude more complexity.

In defining a self-replicating structure the notion of separation between structures needs to be made precise. The three degrees of separation among two structures (or in general, configurations), are noted. Using notation from [8], The set of all non-empty cells in a configuration C is the support function, $\sup C$. Two configurations C and C' are *distinct* [30, pg. 22] if

$$\sup C \neq \sup C' \tag{4}$$

C and C' are *disjoint* if

$$\sup C \cap \sup C' = \emptyset \tag{5}$$

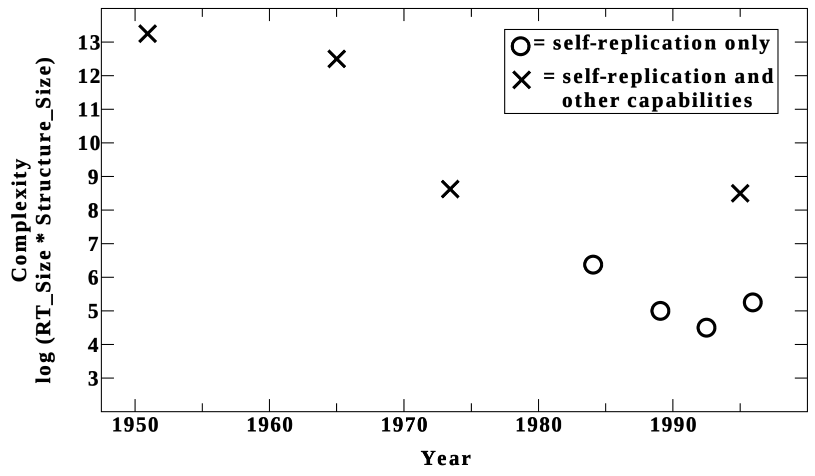


Figure 4: Plot of self-replicating system complexity for cellular automata models. From left to right, the $\times$ symbols are von Neumann [47], Codd [8], Vitányi [45], Perrier *et al.* [34], and the O symbols are Langton [19], Byl [5], Reggia *et al.* [38], Lohn and Reggia [26].

The third and strongest form of separation is called *isolation*. The neighborhood function of a cell α is $g(\alpha)$. The function $g(\alpha)$ generates the set of cells comprising the neighborhood of cell α

$$g(\alpha) = \{\alpha, \alpha + \zeta_1, \ldots, \alpha + \zeta_{n-1}\} \tag{6}$$

where $\zeta_i (i = 1, .., n-1)$ are coordinates relative to α and n is the neighborhood size as defined previously. As an example, the von Neumann neighborhood is expressed as

$$g(\alpha) = \{\alpha, \alpha + (1,0), \alpha + (-1,0), \alpha + (0,1), \alpha + (0,-1)\} \tag{7}$$

which generates the set of five cells: center, top, left, bottom, right.

Let the *neighborhood function of a configuration* C be defined as the set of all cells that are in the neighborhood of C's non-quiescent cells. This function is denoted $G(C)$ and is expressed as

$$G(C) = \bigcup_{\alpha \in \sup C} g(\alpha) \tag{8}$$

A configuration C is *isolated* from configuration C', denoted $C \dashv\vdash C'$, if the set of cells common to both configuration's neighborhoods is not in $\sup C$. This is expressed as

$$\sup C \cap (G(C) \cap G(C')) = \emptyset \tag{9}$$

Figure 5 illustrates with an example the differences between the degrees of separation among configurations.

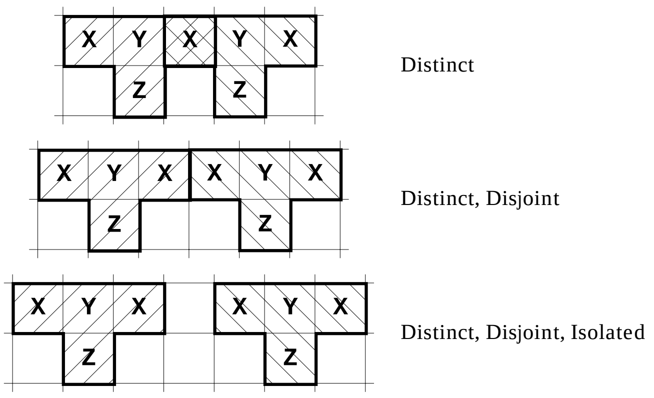


Figure 5: Illustration of the terms distinct, disjoint, and isolated with respect to two example 4-component structures. The von Neumann neighborhood is assumed.

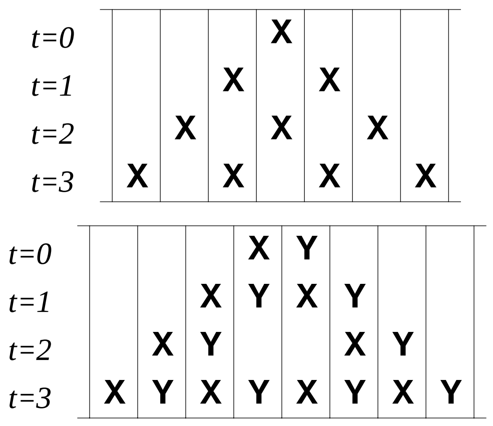
A configuration S is a self-replicating structure if the following criteria are met. First, S is a structure comprised of more than one non-quiescent cell, and changes its shape during its self-replication process. Second, replicants of S, possibly translated and/or rotated, are created in neighbor-adjacent cells by the structure. Third, there must exist a time t such that S can produce i or more replicants, for any positive integer i, for infinite cellular spaces (Moore's criteria [30]). Fourth, if the self-replication process begins at time t, there exists a time $t + \Delta t$ (for finite $\Delta t > 1$) such that the first replicant becomes isolated from the parent structure.

The above requirements encompass the more recently reported models of self-replication, yet they preclude most trivial self-replication processes. They also preclude "artifact" replicants – structures that form the appropriate size and shape, for example, from a supply of unused components without being directed to do so. The issue of triviality was circumvented in early models by requiring universal computation and universal construction. Inspired by biological cells, more recent models (those starting with [19]) have abandoned this requirement by insisting that an identifiable instruction sequence be treated in a dual fashion: interpreted as instructions (translation), and copied as raw data (transcription). As with unsheathed loops, we consider the instruction sequence and the structure itself to be the same, and thus the structure's components directly influence its self-replication process.

Trivial self-replicating structures are easily produced. For example, a 1-D, 3-neighbor CA can be made to give the behaviors shown in Figure 6. In both examples shown, the seed structures are shown at t=0 and replicants subsequently appear isolated. Note that the structure of Figure 6(a) would be trivially self-replicating because of the requirement that a non-trivial self-replicating structure's size be greater than one. Also the requirements above state that self-replication processes must include shape-changing steps. While Figure 6(b) does meet these requirements, it is considered trivial because its shape remains unchanged while all of its components simultaneously split at t=1.

3 Genetic Algorithm

Relatively few studies have reported using genetic algorithms (or related techniques) to automatically produce rule tables for cellular automata (see, for example, [1], [29], [39]). With the exception of our preliminary report [24], there are no reports of using GAs to discover self-replicating structures in cellular space models. Such research has most likely not been undertaken for at least two reasons. First, the computational load can become enormous. Rule tables for modest CA systems can quickly grow extremely large (e.g., 25,000 transition rules for a ten-state, five-neighbor, strongly rotation symmetric CA), and manipulating numerous large rule tables in a GA (even



(a)
(b)

Figure 6: Examples of trivial self-replicating structures in a 1-D, 3-neighbor cellular space model. Seed structures are seen at t=0, and three more time steps are shown. (a) 2-state, 1-component structure; (b) 3-state, 2-component structure.

when compressed) can exhaust the memory capacity and processing capabilities on many computer systems. Second, identification of effective fitness functions is a difficult task. Apparently obvious fitness functions, such as those that count the number of replicants, are useless early on as there will typically be no replicants. This has been borne out in extensive testing of randomly initialized rule tables, and agrees with intuition, given the immense search space sizes. Further, comparing a developing structure to a predefined template of multiple seed structure copies (located in specific positions) by way of pattern matches fails to give partial credit during the replication cycle itself, when the structure has changed shape as it undergoes its self-replication. Having a predefined template also imposes a strong bias on the self-replication process, which is undesirable since it severely limits the types of self-replicating behaviors that could possibly emerge.

Another difficulty involves the temporal aspects of self-replication: at what time step or range of time steps should the quality of self-replication be decided? Using cellular space state data from a single time step would require knowing *a priori* in which configuration replicants will appear and assumes that replicants appear all at once rather than at different time steps. Data from multiple time steps are needed so as to identify replicants as they are produced. This leads to the problem of deciding which configuration to start with, and how many subsequent configurations to examine for self-replicating behavior. In general, assigning small values of fitness to behaviors that do not resemble self-replication yet have the potential to evolve into such a process is a difficult problem. A solution to this problem is one of the key contributions of this chapter.

Since the GA begins with a population of randomized rule tables, it is extremely unlikely that such rule tables will lead to self-replicating behavior. If the fitness functions of the GA assign positive fitness values only to rule tables that lead to self-replicating behavior, then all rule tables will have fitnesses of zero, and the GA will not be able to apply its genetic operators effectively. In such cases the search degenerates into an ineffective random search process. Assigning small values of fitness to behaviors that do not resemble self-replication yet have the potential to evolve into such a process is needed for an effective search.

The fitness functions derived in this section are general in that they could be used in other 2-D cellular space automata models, and any size and shape seed structure containing unique components may be used. In addition, the fitness functions do not impose a strong bias toward any particular process of self-replication. That is, in their definitions, the fitness functions do not assign fitness based on aspects such as: 1) the contents of specific cell locations at specific instants, 2) whether/how the structure should translate or rotate itself over time, 3) the quantity/timing of replicant production, or 4) the extent to which configurations match a predefined configuration.

The problem of automatically finding rule tables that yield self-replicating structures in cellular automata is a type of rule discovery problem. An overview of our specific approach is given in Figure 7. After the rule discovery process has produced a candidate solution, the cellular space must be manually simulated to determine if the discovered rule table does result in a self-replicating structure. This step is needed since a rule table that scores the highest fitness could potentially be trivial or circumvent the fitness function in unanticipated ways.

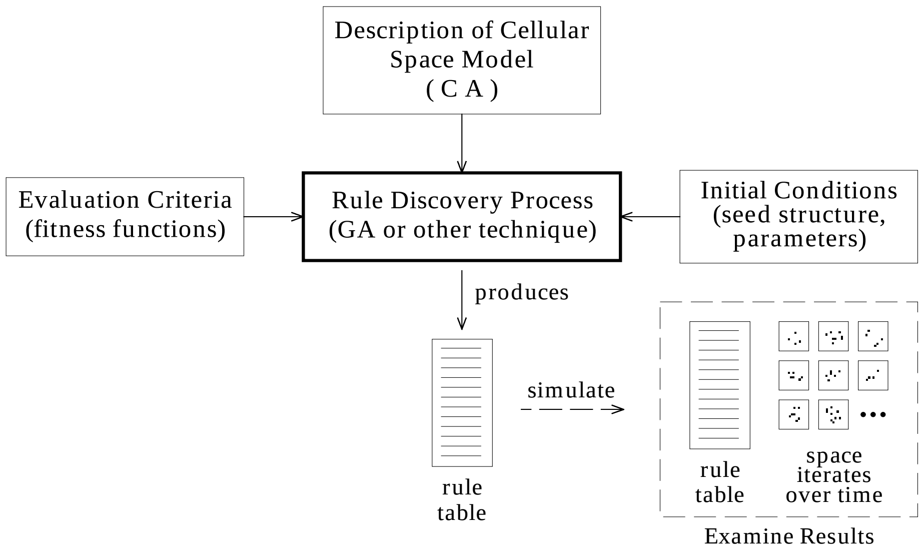

Description of Cellular Space Model (C A)
Evaluation Criteria (fitness functions)
Rule Discovery Process (GA or other technique)
Initial Conditions (seed structure, parameters)
produces
rule table
simulate
rule table
space iterates over time
Examine Results

Figure 7: Overview of the rule discovery system showing the major components, production of a discovered rule table, and the manner in which the discovered set of rules is analyzed.

3.1 Evolving Rule Sets using a GA

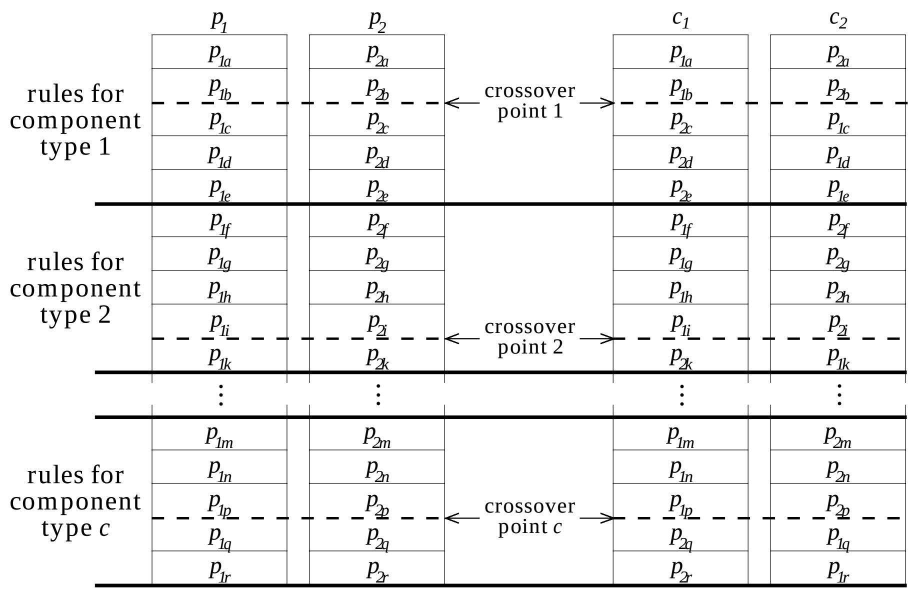
The genetic algorithm applied in this study employed a small population size of 100 rule tables for two reasons: computer resource limitations, and for consistency so that GA performance using larger CA models[2] could be compared directly to those that had small CA models. Rule tables in the GA were encoded in a natural manner: a table containing next-state entries indexed by neighborhood patterns (four rule tables are shown in Figure 8). For example, if the

[2]Increased number of states, thus larger rule tables.

ith transition rule of the rule table were `BCCDE` $\rightarrow$ `A`, then the ith entry in the encoded rule table would be `A`. Rule table sizes for a system with three component types are 14787, and 838 for CA models with orientation sensitive and orientation insensitive input, respectively. The type of crossover used here was a version of multi-point crossover whereby single-point crossover is applied within segments marked by the heavy lines in Figure 8. A crossover point was randomly selected within each segment, and single-point crossover occurred in each segment. The diagram shows a CA model where each component type has only five transition rules for illustration purposes. Using "`d`" to represent a don't-care cell-state, we can imagine that transition rules of the form `Xdddd` $\rightarrow$ `d` program the behavior of component `X`, `Ydddd` $\rightarrow$ `d` program the behavior of component `Y`, etc. Performing crossover within each segment allows the GA to operate on the behavior of each component individually. At a higher level, because the fitness functions are rewarding cooperation among components, component types are evolved together in a co-adapted manner. Empirical results comparing this crossover technique to that of single-point crossover (across the entire rule table) showed higher performance for the multiple application of crossovers. After selection and crossover, each transition rule was subject to mutation which occurred by randomly choosing a new state.

The evaluation phase of the GA is depicted in Figure 9. Evaluating each individual requires that a complete CA simulation be executed (Figure 9, middle). The initial conditions for each evaluation were comprised of a rule table and seed structure. The seed structures remained fixed in every GA evaluation, and the specific structures used in the experiments described below were comprised of the two, three, and four unique components as shown in Figure 10.

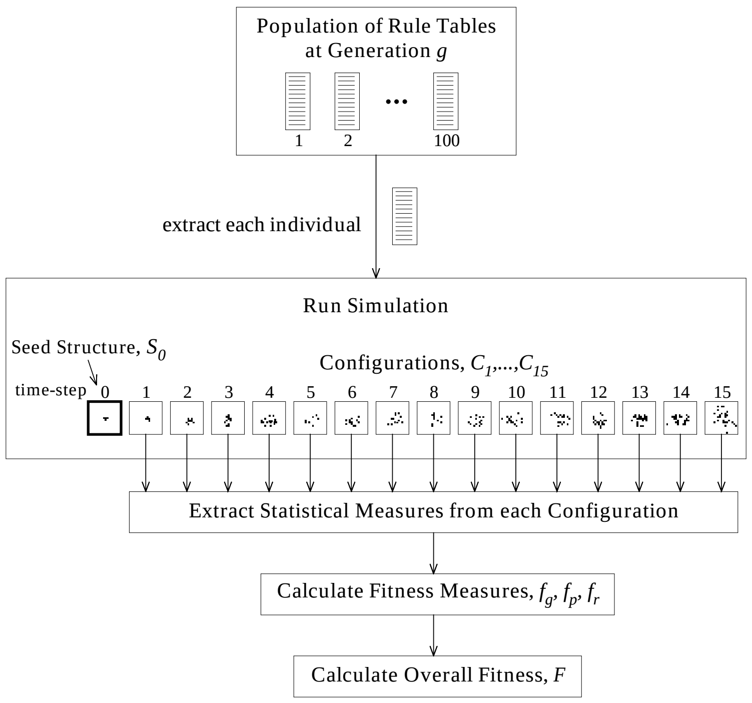
As shown in Figure 9, the fitness function used data extracted from the first 15 configurations during an individual evaluation. Since the seed structures that we dealt with were very small, fast replication cycles were very likely [38]. Such cycles were generally less than 10 time steps, with critical steps of the self-replication process occurring very early on, generally in the first five time steps. Therefore, so as not to exclude any useful information that could occur early on,



p_1
p_2
c_1
c_2
rules for component type 1
rules for component type 2
rules for component type c
crossover point 1
crossover point 2
crossover point c

Figure 8: Illustration of crossover using rule tables. Parents p_1 and p_2 (left) are recombined to form offspring c_1 and c_2 (right) by segmenting the rule tables into c partitions according to component type, and crossing over transition rules within each partition, with each crossover point chosen at random.

statistics from the first time steps are included. The choice of how many configurations to use in the evaluation of a self-replication process was determined as follows. Let Δt denote the duration of time which will be examined for fitness calculations. If Δt is too small, this may not give enough time for a self-replicating process to emerge. If Δt is too large, two undesirable situations will arise. First, the efficiency of the GA will decrease since the GA will be spending more time examining behaviors that, in general, do not exhibit self-replication. The CA simulations are inside two loops of the GA: one for each population member and one for each generation. The product of these two numbers is on the order of 200,000 for our experiments. Thus the expression $200,000\Delta t$ represents the total



Population of Rule Tables at Generation g
1
2
100
extract each individual
Run Simulation
Seed Structure, S_0
Configurations, $C_1,\ldots,C_{15}$
time-step
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Extract Statistical Measures from each Configuration
Calculate Fitness Measures, f_g, f_p, f_r
Calculate Overall Fitness, F

Figure 9: Evaluation phase of genetic algorithm.

number of CA simulation time steps executed during a run of the GA. For statistical sampling purposes, we required 100 GA runs per experiment. Therefore each increment to Δt adds 20,000,000 more time steps to the overall experiment, which becomes a significant computational burden. Second, as Δt increases, the likelihood of coincidental seed structure copies appearing increases, and this could potentially disrupt fitness function calculations. Such copies could then inflate the fitness values and hinder the search process. Based on previous studies of hand-designed self-replicating structures [38] and considering these tradeoffs, a value of $\Delta t < 10$ was determined too restrictive and $\Delta t > 20$ too large. A value of $\Delta t = 15$ time steps was chosen.

Statistics collected from the 15 configurations of each CA simulation are time-averaged component counts (multiplicities), adjacency information, and replicant counts. Multiplicity values M_v^t record the

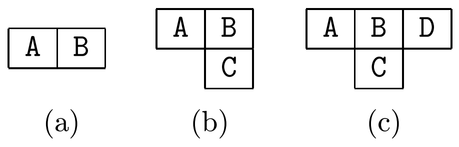


Figure 10: Seed structures: (a) 2-component; (b) 3-component; (c) 4-component.

quantity of each component type v at each time t. Adjacency information includes relative positioning data regarding each component type over time. Replicant count quantifies the number of replicants seen at each time step.

After collection of these three types of statistics, three corresponding fitness measures are computed and combined in an overall fitness function F for each simulation. The first is a growth measure, denoted f_g, which correlates growth in number of individual component types with high performance. The second criteria is called the relative position measure, denoted f_p. This measure is concerned with awarding fitness to component types that have a high percentage of neighboring components positioned in the same manner as is seen in the seed structure. The third criteria is one that measures isolated replicants, denoted f_r. This function scans configurations looking for isolated replicants and awards proportionate amounts of fitness depending upon the number of replicants seen over time. Isolated means that a structure separates completely and is surrounded by only quiescent cells, as explained earlier.

The fitness function F used in the selection process is a linear sum of the above three measures. Specifically, defining a fitness measure vector $\mathbf{f} = (f_g, f_p, f_r)$ and a weight vector $\mathbf{w} = (w_g, w_p, w_r)$, we have

$$F = \mathbf{f} \cdot \mathbf{w} \tag{10}$$

For convenience, the fitness measure functions in $\mathbf{f}$ are each normalized to values in $[0, 1]$, and weights (positive) are such that $w_g + w_p + w_r = 1$.

3.2 Fitness Measures

In order for a self-replicating process to emerge, one would expect to observe, over time, increasing quantities of the individual components. In analyzing past, manually-designed self-replicating structures, it was seen that individual component counts, or multiplicities, generally increased over time, punctuated by periods of plateaus and small decreases in value.

The growth measure f_g used computes the degree to which each component type maintains an increasing supply of components from one time step to the next. The number of components of type v_i at time t is denoted $M_{v_i}^t$. Multiplicity data forms a $\tau \times c$ table, since τ time steps are used and c components are present in the simulation:

$$\begin{array}{cccc} M_{v_1}^1 & M_{v_2}^1 & \cdots & M_{v_c}^1 \\ M_{v_1}^2 & M_{v_2}^2 & \cdots & M_{v_c}^2 \\ \vdots & \vdots & \ddots & \vdots \\ M_{v_1}^\tau & M_{v_2}^\tau & \cdots & M_{v_c}^\tau \end{array}$$

To distill these values into a single meaningful value, multiplicities are first converted using a function ρ_v, which assigns fitness based on whether a given component type increased its production or stayed the same, and no fitness if it decreased:

$$\rho_v(t) = \begin{cases} 1 & \text{if } M_v^t > M_v^{t-1} \\ 0.5 & \text{if } M_v^t = M_v^{t-1} \\ 0 & \text{if } M_v^t < M_v^{t-1} \end{cases} \qquad 0 < t \leq \tau \tag{11}$$

For example, if there were 12 `Y` components at $t = 5$ and 14 at $t = 6$, then $\rho_{\mathtt{Y}}(t = 6)$ would be assigned a value of 1. The function ρ encourages increased quantities of components from one time step to the next, but does not harshly penalize fitness when production declines.

The growth measure f_g is then calculated by summing all ρ_v values and then dividing by the maximum attainable fitness

$$f_g = \frac{1}{\tau c} \sum_{v \in V} \sum_{t=1}^{\tau} \rho_v(t), \tag{12}$$

so f_g calculates a measure of how well the supply of all components increased over time. One might propose simply using a function that assigns high fitness when the total component count increases over time. However, since this does not distinguish among individual component types, such a function could encourage growth of only one or possibly two components, as this will satisfy such a function.

The relative position measure f_p is the most important fitness measure of the three presented here. If a rule table is approximating support for self-replication, it would be expected that an individual component would frequently find itself surrounded by the same components that surrounded it in the seed structure. The function f_p measures the degree to which such relative positionings are satisfied over time. Note that correct relative positions do not necessarily have to occur simultaneously (i.e., during the same time step) among the components of the structure in order for partial fitness to be awarded. The ability of f_p to give partial fitness in this manner proved to be critical to providing the GA with positive reinforcement needed to search effectively.

The seed structure plays a critical role in deriving the function f_p since it contains the relative positioning information. The adjacencies contained in the seed structure are formulated in terms of an adjacency vector, $\mathbf{s} = (s_{v_1}, s_{v_2}, \ldots, s_{v_c})$, representing the number of neighborhood-adjacent components for each component type. Here s_{v_i} represents the number of components that are neighborhood-adjacent to component v_i in the seed structure. Examples of adjacency vectors are shown in Figure 11.

The function $m_v(t)$ is the number of neighbors of component v at time t that were the same type and in the same relative position as in the seed. The function $\sigma_v(t)$ represents to what degree, at time t, all the components of component type v have the same neighbors

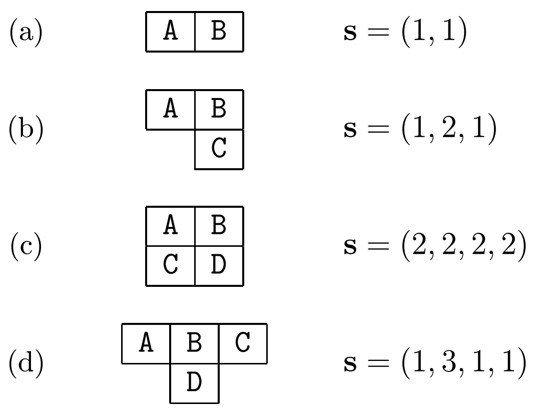


Figure 11: Examples illustrating the adjacency vector of various seed structures.

as in the seed and is defined as:

$$\sigma_v(t) = \begin{cases} 0 & \text{if } M_v^t \leq 1 \\ \frac{m_v(t)}{M_v^t \, \cdot \, s_v} & \text{if } M_v^t > 1 \end{cases} \tag{13}$$

When $M_v^t \leq 1$, component v is extinct or is presumably part of the seed. When $M_v^t > 1$, $\sigma_v(t)$ is the ratio of $m_v(t)$ to the maximum value possible. Thus $\sigma_v(t)$ can never exceed unity because $m_v(t)$ represents the number of adjacent cells that matched correctly, and the denominator represents the total number of adjacent cells. As in the growth fitness measure, a $\tau \times c$ table of values is generated by σ. Measure f_p is then defined to be the mean of $\sigma_v(t)$ over component types and time, weighted by $\mathbf{s}$ as follows:

$$f_p = \frac{1}{\tau \sum_{v \in V} s_v} \sum_{v \in V} \sum_{t=1}^{\tau} s_v \sigma_v(t) \tag{14}$$

The adjacency vector $\mathbf{s}$ as used in Eq. (14) gives higher priority to components that have more neighbors in the seed structure. For example, the `B` component in Figure 11(d) receives a normalized weight of 0.5 and the other components each receive 0.17.

Lastly, the isolated replicant fitness measure f_r correlates fitness with increasing numbers of isolated replicants formed during the course of a simulation. In contrast to the relative position fitness measure, f_r provides little if any positive reinforcement to the GA during the beginning of the discovery process since no replicants are typically present. Its main purpose is to guide the GA toward fitter and fitter self-replicating structures once nascent ones have been discovered.

Let r_t represent the number of isolated replicants in configuration C_t. Then f_r is calculated as a sigmoid/logistic function of the maximum r_t obtained in the simulation:

$$f_r = \frac{1}{1 + e^{-(\max(r_t) - \theta)}} \tag{15}$$

The constant θ represents the inflection point of the sigmoid, and a value of 4.0 was typically used. This allows fitness to increase at a faster rate during periods when two or three isolated replicants are seen, and at a slower rate when more than four appear.

The GA was iterated over many generations until either the best of generation individual achieved a fitness greater than 0.9, or until it reached a prespecified number of generations. The main GA parameters used were: population size of 100, crossover rate between 0.6 and 0.8, mutation rate between 0.08 and 0.10, and a maximum of 2000 generations.

An overview of the technique is depicted in Figure 12. The approach taken here is to execute numerous independent GAs, and use statistical methods to analyze the results of the set of GAs. As it is used here, one "experiment" is taken to be a set of 100 trials, with each trial being an identical GA except that the stream of random numbers differs from one instance to the next. The top box of Figure 12 depicts the common inputs to all of the independent GAs. While executing, each GA stores the highest-fitness rule table it has ever evaluated, and stops when the convergence criteria are met. At that point, the highest-fitness rule table is its output (Figure 12, middle). The outcome of each trial is either success (a self-replicating structure found) or failure. Such a decision must be made by manual examination of a subsequent simulation, since the rule table with the highest fitness value may not always conform to requirements. The quantity of self-replicating structures found divided by 100 (trials) is called the *yield*. The goal of a given experiment is to maximize the yield.

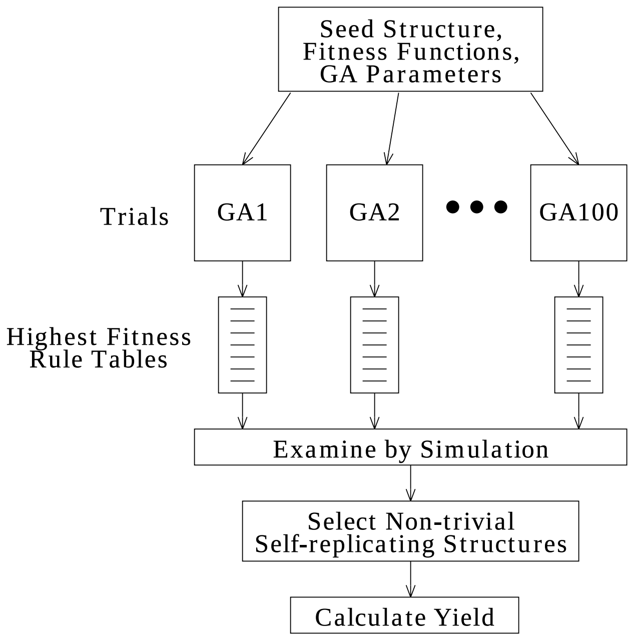


Figure 12: Overview of experimental method.

We calculated the statistical significance of the yields obtained from each experiment. Comparing the yield found using the genetic algorithm in an experiment to the yield found via random search allowed us to gauge the effectiveness of the search process. For every experiment that was run, comparable trials using random search were also done. In each trial of random search, zero self-replicating structures were produced. Applying Fisher's Exact test, let d represent the number of replicants discovered by the GA. The 2×2 table can be written:

	# successes	*# failures*
GA	d	$100 - d$
Random Search	0	100

We use this statistical test to calculate the statistical significance for our results in the next section.

4 Experimental Results

The experiments conducted can be classified according to the size of the seed structure used. With the computational resources available, structures having two, three, or four components were feasible to use in the experimental framework described above. For example, experiments using four-component structures required approximately one week of dedicated time on a 40-node processor farm consisting of 275 MHz DEC Alpha processors. The limitations on this resource allowed for only three experiments to be conducted using four-component seed structures.

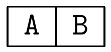
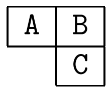
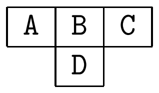
For each of the seed structures (Figure 10), two variations of cellular automata models were used. We call the CA model using orientation sensitive input the "standard" CA model, since it is identical to what has been used in research to date. We included it because it has been studied the most with respect to self-replicating structures and it is desirable to see how it performs compared to CA models that use orientation insensitive input.

In the context of the fitness calculation of F (Eq. (10)), it is desired to optimize F by finding an ideal vector $\mathbf{w}$ with which to weight the independent fitness measures f_g, f_p, and f_r. A second meta-level GA [11] was used to find this weight vector. Under the control of the meta-GA, the primary GA was executed repeatedly. Since the primary GA was resource intensive by itself, experimenting with the meta-GA required that smaller GA parameters be used. A population of 20 14-bit individuals was used, with each individual encoding weights w_1, w_2, w_3. The fitness function employed for the meta-GA was the f_r fitness measure described above. Thus if a primary GA run was able to find isolated replicants, this would give high fitness to a specific weight vector, which would then be bred into the next population of the meta-GA. A thorough study of weight vector optimization was deemed prohibitive, but the weight vectors found in about 10 meta-GA runs generally produced better results than those found during manual trial and error.

4.1 Production of Replicants

Table 1 presents the yields of self-replicating structures found during 100 trials for the CA models and seed structures studied. The labels CA_{oii} and CA_{osi} denote the cellular automata model using orientation insensitive input and orientation sensitive input, respectively.

Table 1: Highest numerical yields from each experiment.

Seed Structure	Names		Yields	
	CA_{oii}	CA_{osi}	CA_{oii}	CA_{osi}
	PS2WI9V	PS2W9V	0.93	0.49
	PS3WI13V	PS3W13V	0.22	0.03
	PS4WI17V	PS4W17V	0.02	0.00

Beginning with the 2-component structures, the model with orientation insensitive input had the most successful results with 93 discovered self-replicating structures. While each of the 93 rule tables are distinct, many of the self-replication processes were qualitatively similar.

For the 3-component experiments, it is seen that the CA model with orientation insensitive input again had the highest yield. The orientation sensitive input CA model had a yield of only 3%, which is not statistically significant at the 0.05 level of significance using Fisher's Exact test. Three-component yields were lower than that for 2-components, indicating that the discovery process is more difficult for larger structures. This agrees with the intuition that self-replication processes for 3-component structures are more complex than for structures having two components.

In the 4-component experiments, the CA$_{oii}$ model had the only non-zero yield. Although not statistically significant at the 0.05 significance level, it is of interest that the GA was able to discover 2 self-replicating behaviors in only the CA$_{oii}$ model, the model which gave the best results in the other experiments.

These results suggest that by using the orientation insensitive input paradigm, higher yields of self-replicating structures may be found. One of the potential reasons for the higher yields in the CA$_{oii}$ model is that it has the smallest search space size of the two models, and thus the GA may have a slightly easier search task. To quantify the correlation between increasing yields and decreasing search space sizes, we calculated the sample correlation coefficient r for the three seed structures shown in Table 1: $r = -0.237$ for the 2-component seed, $r = -0.406$ for the 3-component seed, and $r = -0.499$ for the 4-component seed. All coefficients are negative, indicating that as the search space search increases, the yields decrease. With values of r ranging between -0.2 and -0.5 we can posit that there is some degree of correlation, but not strongly so.

4.2 Discovered Structures

Representative samples of the automatically discovered self-replicating structures are shown in this section, and are of interest in themselves in that they have significant differences from past manually-designed self-replicating structures [5], [19], [38]. A naming convention is established so that each structure can be given a unique name and the underlying cellular space model can be easily identified. The notation in [38] is used, augmented with the prefix `PS` to indicate polyomino structures, and the letter I is added prior to the number of states field to denote orientation insensitive input. Table 1 lists six structures using this naming convention.

Figs. 11–13 present the first nine time steps of configurations for two typical 3-component structures and a 4-component structure. These structures differ from previously reported hand-coded structures in unanticipated ways. Most striking is the way parent structures move while replicating. In addition, instead of "dead structures" forming inside of an expanding colony of replicants (common phenomena in previous work), we see that less organized "debris" forms due to collisions of moving structures (see Figure 16).

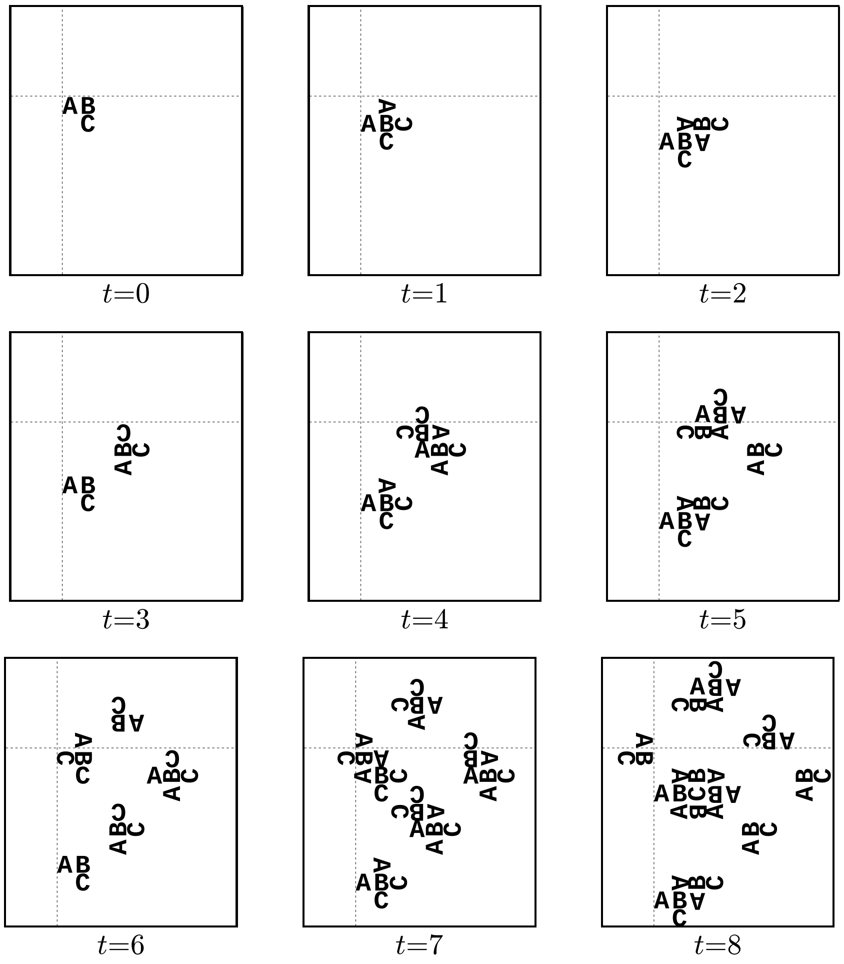


Figure 13: Self-replicating structure `PS03WI13V`. The seed structure moves downward on successive time steps, producing rotated replicants. A 5-step replication process can be seen, and production of the second replicant (t=4) begins while the first replicant is still forming. Thus the first isolated replicant appears at t=5 and the second at t=6. Each replicant is rotated and forever moves in a straight line producing rotated replicants of its own.

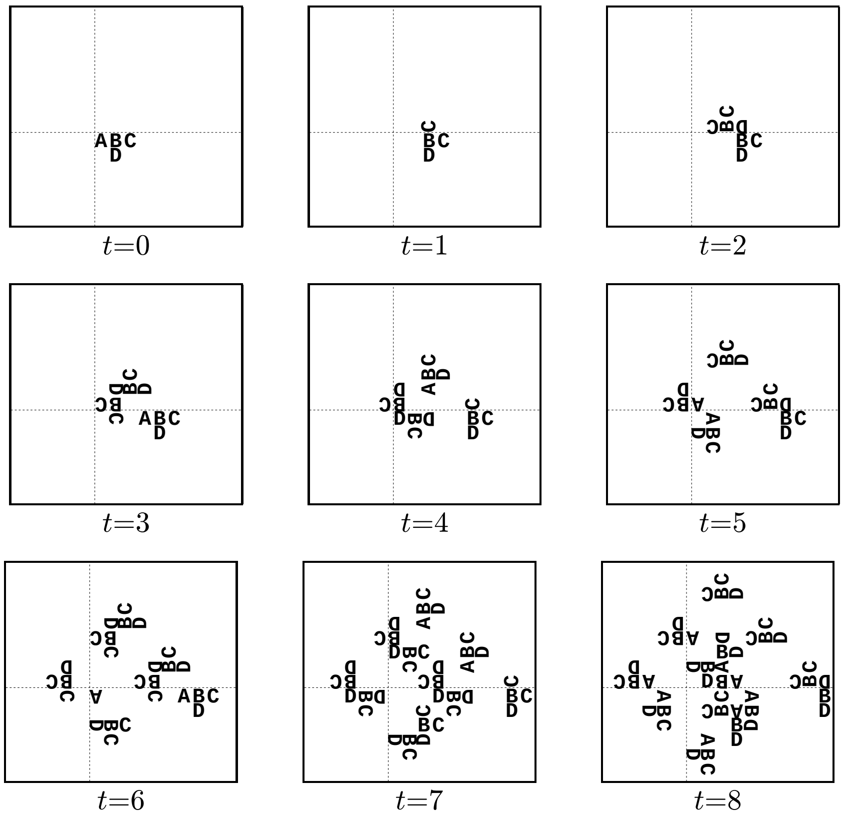


Figure 14: Self-replicating structure `PS04WI17V`. The seed structure moves towards the right on successive time steps and produces two replicants: the first is seen at t=4 and then again at t=7 along with the second replicant (upper right quadrant of each respective frame). These replicants are rotated 90° counterclockwise and proceed upward. During the production of the first replicant, debris forms (near coordinate system origin of t=3 and t=4) and coalesces into two structures seen at t=5, lower left. One structure moves downward and attempts to self-replicate but due to crowding, is unable. The other moves to the left and produces its first replicant at t=8 (lower left quadrant).

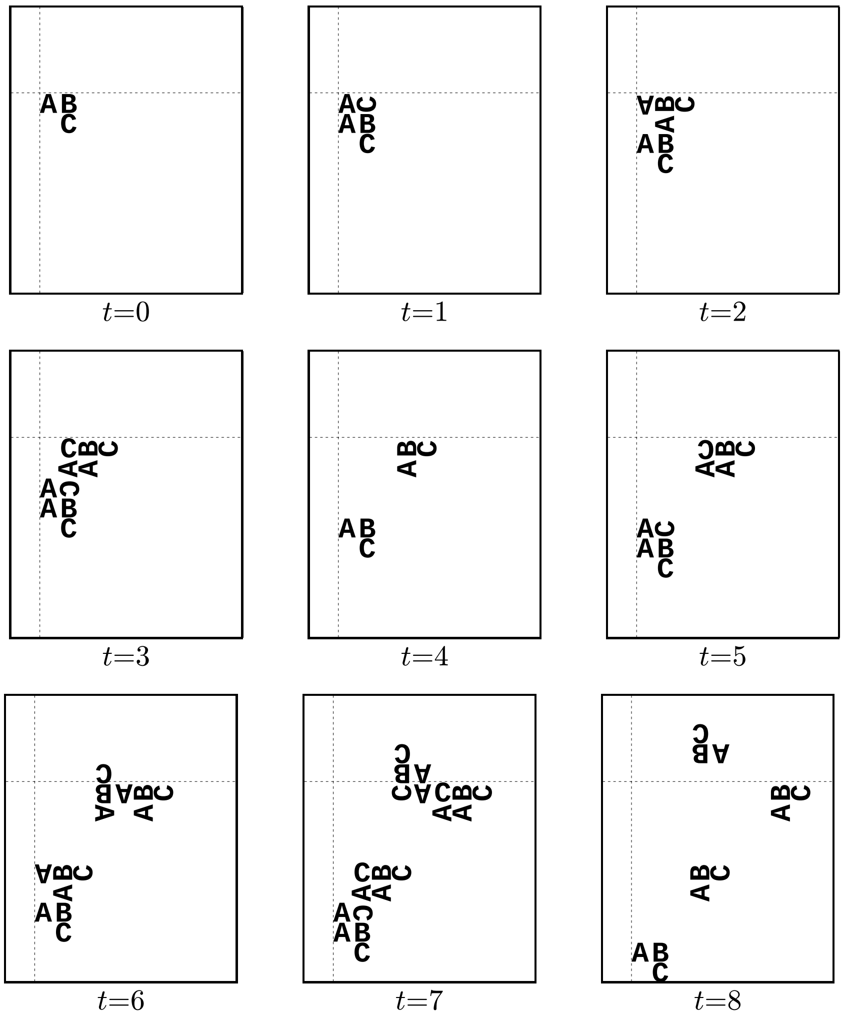


Figure 15: Self-replicating structure `PS03W13V`. The seed structure proceeds downward while producing an isolated replicant every four time steps. Note that the first replicant is fully formed at t=2, yet not isolated. A unique behavior seen in this structure is the fact that there are no unused components during much of the colony formation (later, the colony collapses in on itself and collisions occur).

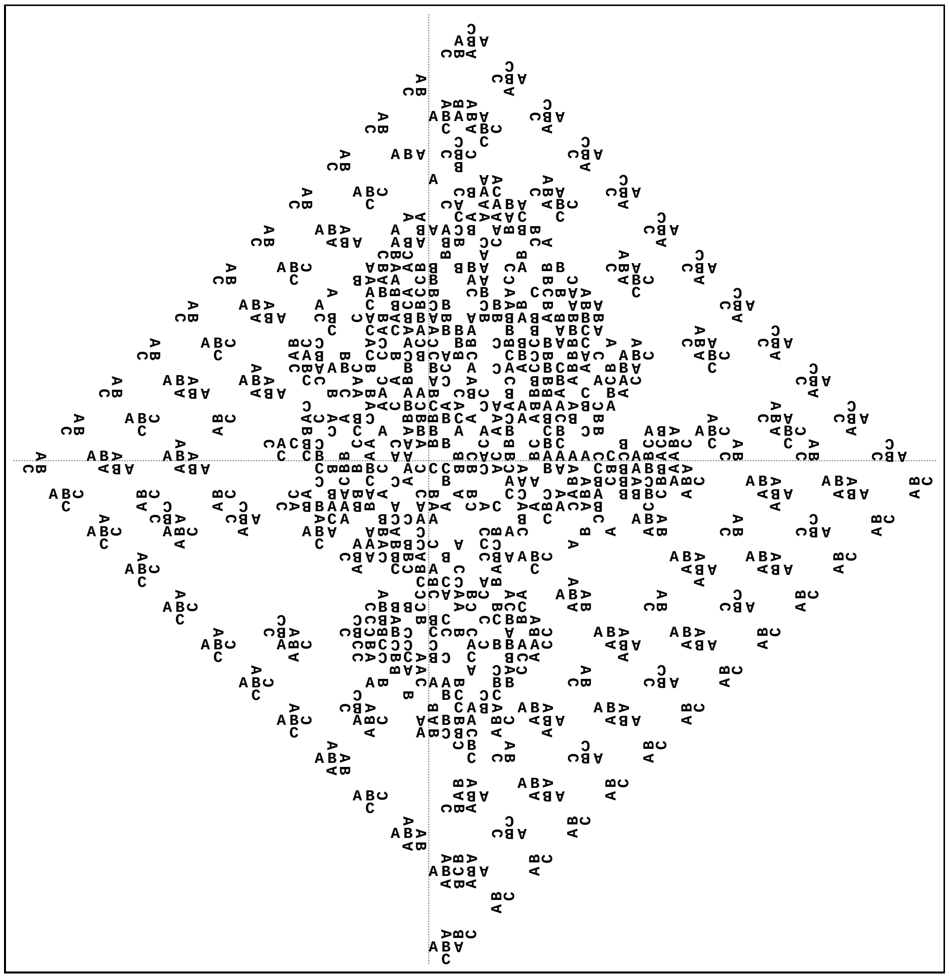


Figure 16: Self-replicating structure PS03WI13V at time step 38 illustrating formation of debris inside of the expanding colony.

4.3 Statistical Testing

Employing Fisher's Exact test as described in the previous section, it is relatively easy to show that when there are four successes (four self-replicating structures discovered in 100 trials), p=0.061, and with five successes, p=0.029. Thus a yield of five or more self-replicating structures is statistically significant at the 0.05 significance level. For the experimental results presented in Table 1, it is seen that some yields are not statistically significant at the 0.05 significance level. For example, in the 3-component experiments, while the 22% yield is statistically significant, the 3% yield from the CA$_{osi}$ model is not. Also, both of the 2-component experiments and none of the 4-component experiments gave statistically significant results at the 0.05 significance level.

4.4 GA Performance

A typical GA performance graph showing the behavior of individual fitness measures is shown in Figure 17. This GA run produced a self-replicating structure using $F = 0.05f_g + 0.75f_p + 0.20f_r$. In this case the overall fitness value for a rule table mainly comes from the relative positions of components. Less important are the isolated replicants and growth of components. However, there is a deeper interpretation. The apparently insignificant growth measure plays a key role in the early generations of the GA, the relative position subsequently maintains a steady increase in F, and the isolated replicant measure serves to lock-in a newly discovered self-replicating structure. These behaviors suggest that the three parts of the fitness function support each other in complex and unanticipated ways.

Figure 17 shows values of F, f_g, f_p, and f_r for the highest ranking rule table of each generation. The growth measure f_g is seen to be the most volatile, and this agrees with intuition: since it has the smallest weight in F, larger fluctuations are easily tolerated and have less of an effect on F. The relative position measure f_p remains the highest contributor in most generations, which is not surprising since it has 75% of the weight in F. The isolated replicant measure f_r, being the hardest fitness measure to satisfy, remains near zero for many generations until the other measures discover a rule table that promotes elements of a self-replicating process.

As can be seen in Figure 17, during the first few generations, the growth measure increases rapidly, helping to prime the search. The growth measure is the easiest way of gaining fitness since it is only concerned with quantities of components and not positioning. The relative position measure increases more slowly, as increasing numbers of components gain proper positioning. Near generation 150, isolated replicants begin to proliferate. The isolated replicant measure then increases quickly and the GA converges, with only small refinements being made.

Typically performance curves exhibit two regions of behavior, as seen in Figure 17. The first region is characterized by high component growth: $f_g > f_p > f_r$. The performance enters the second region when the isolated replicant measure dominates, indicating convergence. In this region we have: $f_r > f_p > f_g$. Since growth is weighted the least, it has less of an effect when the other measures are elevated.

A simple experiment was constructed to determine if each of the three fitness measures, by themselves, can promote development of a self-replicating structure. If none of the individual fitness measures is able to produce such a structure, this suggests that the emergence of self-replication is dependent on the interactions and combined effects of the individual measures. The three experiments involve setting the weight vector $\mathbf{w} = (w_g, w_p, w_r)$ as $\mathbf{w} = (1, 0, 0)$, $\mathbf{w} = (0, 1, 0)$, or $\mathbf{w} = (0, 0, 1)$. With each of these three weight vectors, no self-replicating structures were ever discovered in 100 GA runs (using 3-component seed structures), suggesting that the fitness measures interact and depend on each other to promote self-replicating behaviors.

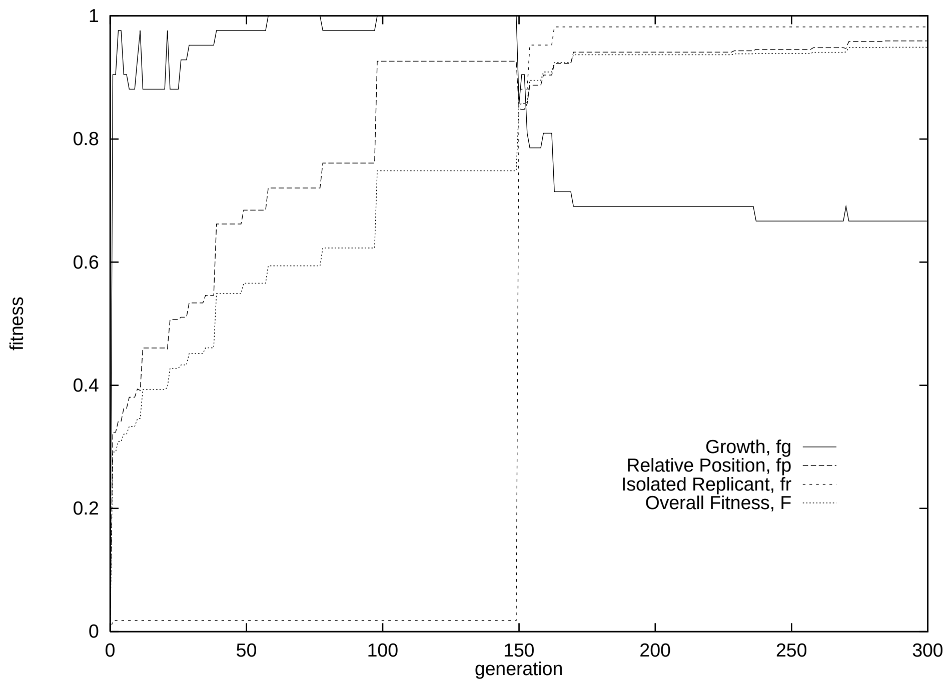


Figure 17: Individual fitness measure values for the best-of-generation rule table during GA discovery of `PS03WI13V`. The overall fitness function is $F = 0.05f_g + 0.75f_p + 0.20f_r$.

5 Discussion

Our experiments show, for the first time, that rule sets for self-replicating structures can be automatically discovered. Creating rules for self-replicating structures is difficult and has only been done manually in the past. Prior to this work, fewer than 30 hand-designed self-replicating structures have been reported in 45 years.

This chapter presented three main contributions towards automating the discovery of self-replicating structures. First, we demonstrated that the process of discovering self-replicating structures in cellular automata can be automated. The simulation results showed that statistically significant amounts of such structures were generated. The discovered structures compared favorably in terms of simplicity with those generated manually. In addition, the structures did not rely on extraneous components as seen in past models. The process of replication discovered in many cases was quite interesting because it differed in unexpected ways from previous manual attempts, e.g., the structures all move during self-replication.

Second, we presented a multiobjective fitness function that was able to promote diverse self-replicating behaviors. The derived fitness functions are general in the sense that they can be used with other reinforcement learning techniques. Also, these fitness measures do not impose undue biases toward any particular process of self-replication as evidenced by the large variety of specific rule tables found.

Third, a new variation of cellular automata was presented which produced a higher yield of self-replicating structures, yet maintained the desirable properties of the underlying cellular model. The technique of orientation-insensitive input is also applicable to other cellular space models and is less demanding of computer resources. We hypothesize that the discovery process is facilitated by having reduced search space sizes under this input method.

These results suggest that further exploration in the space of possible self-replicating structures could yield numerous new structures. The technique presented is independent of structure size, therefore automatic production of larger self-replicating structures is feasible for future study. Finally, this research sheds light on the process of creating self-replicating structures, potentially leading to future studies on the discovery of novel self-replicating molecules and self-replicating assemblers in nanotechnology.

6 Future Work

This chapter presented a successful application of genetic algorithms in automatically designing self-replicating structures. In conjunction with numerous previous studies, we have provided further evidence that techniques from evolutionary computing are effective at finding solutions in large design spaces. Following this theme into the realm of electronic design, researchers have used evolutionary computing to automatically design circuits [18], [27] that are feasible to construct. It is quite possible that, in the future, a subset of electronic system design may be routinely accomplished using evolutionary computing techniques. Even farther ahead lies the application of these techniques to designing nanoelectronic structures using carbon nanotubes, an area which is currently being investigated by the authors.

Acknowledgment

This work was supported by NASA Award NAGW-2805.

References

[1] Andre, D., Bennett, F. H., and Koza, J. R. (1997), "Evolution of Intricate Long-distance Communication Signals in Cellular Automata using Genetic Programming," in *Artificial Life V*, C. G. Langton and K. Shimohara, Eds. Cambridge, MA: MIT Press.

[2] Arbib, M. A. (1966), "Simple Self-Reproducing Universal Automata," *Information and Control*, vol. 9, pp. 177–189.

[3] Brooks, R. A and Maes, P. Eds. (1994), *Artificial Life IV*, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, MIT Press.

[4] Burks, A. (1970), *Essays on Cellular Automata*, Univ. of Illinois Press.

[5] Byl, J. (1989), "Self-Reproduction in Small Cellular Automata," *Physica D*, vol. 34, North-Holland, pp. 295–299.

[6] Chou, H. H. , Reggia, J. A. (1997), "Emergence of Self-Replicating Structures in a Cellular Automata Space," *Physica D*, in press.

[7] Chou, H. H., Reggia, J.A. (1997), "Problem Solving During Artificial Selection of Self-Replicating Loops," *Physica D*, in press.

[8] Codd, E. F. (1968), *Cellular Automata*, Academic Press.

[9] Drexler, K. E. (1989), "Biological and Nanomechanical Systems: Contrasts in Evolutionary Capacity," in [21], pp. 501–519.

[10] Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley.

[11] Grefenstette, J. (1986), "Optimization of Control Parameters for Genetic Algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, No. 1, pp. 122–128.

[12] Hall, M. (1967), *Combinatorial Theory*, Waltham, MA: Blaisdell Publishing,.

[13] Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: Univ. of Michigan Press.

[14] Holland, J. H. (1976), "Studies of the Spontaneous Emergence of Self-Replicating Systems Using Cellular Automata and Formal Grammars," in *Automata, Languages, Development,* A. Lindenmayer and G. Rozenberg, Eds., North-Holland, pp. 385–404.

[15] Hong, J.-I., Feng, Q., Rotello, V., and Rebek, J. (1992), "Competition, Cooperation, and Mutation: Improving a Synthetic Replicator by Light Irradiation," *Science*, vol. 255, pp. 848–850.

[16] Kephart, J. O. (1994), "A Biologically Inspired Immune System for Computers," in [3], pp. 130–139.

[17] Koza, J.R. (1994), "Artificial Life: Spontaneous Emergence of Self-Replicating and Evolutionary Self-Improving Computer Programs," in [23], pp. 225–262.

[18] Koza, J.R., Bennett, F.H., Andre, D., Keane, M.A., Dunlap, F. (1997), "Automated Synthesis of Analog Electrical Circuits by Means of Genetic Programming," *IEEE Trans. on Evolutionary Computation*, vol. 1, no. 2, July, pp. 109–128.

[19] Langton, C.G. (1984), "Self-Reproduction in Cellular Automata," *Physica D*, vol. 10, pp. 135–144.

[20] Langton, C.G.(1986), "Studying Artificial Life with Cellular Automata," *Physica D*, vol. 22, pp. 120–149.

[21] Langton, C. G., Ed. (1988), *Artificial Life*, Santa Fe Institute Studies in the Sciences of Complexity, vol. VI, Addison-Wesley.

[22] Langton, C. G., Taylor, C., Farmer, J.D., and Rasmussen, S., Eds. (1991), *Artificial Life II*, Santa Fe Institute Studies in the Sciences of Complexity, vol. X, Addison-Wesley.

[23] Langton, C.G., Ed. (1994), *Artificial Life III*, Santa Fe Institute Studies in the Sciences of Complexity, vol. XVII, Addison-Wesley.

[24] Lohn, J.D., and Reggia, J.A. (1995), "Discovery of Self-Replicating Structures using a Genetic Algorithm," *1995 IEEE International Conference on Evolutionary Computing*, Piscataway, NJ: IEEE Press, pp. 678–683.

[25] Lohn, J.D. (1996), "Automated Discovery of Self-Replicating Structures in Cellular Space Automata Models," Dept. of Computer Science Tech. Report CS-TR-3677, Univ. of Maryland at College Park.

[26] Lohn, J.D., and Reggia, J.A. (1997), "Automatic Discovery of Self-Replicating Structures in Cellular Automata," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 3, pp. 165-178.

[27] Lohn, J.D., Colombano, S.P. (1998), "Automated Analog Circuit Synthesis using a Linear Representation," *Proc. of the Second Int'l Conf on Evolvable Systems: From Biology to Hardware*, Springer-Verlag, Berlin, pp. 125-133.

[28] McMullin, B. (1992), "The Holland α-Universes Revisited," in *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life,* F. Varela and P. Bourgine, Eds., Cambridge, MA: MIT Press, pp. 317–326.

[29] Mitchell, M., Hraber, P.T., and Crutchfield, J.P. (1993), "Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations," *Complex Systems*, vol.7, no. 2, pp. 89–130.

[30] Moore, E.F. (1962), "Machine Models of Self-Reproduction," *Proc. Fourteenth Symp. on Applied Mathematics*, pp. 17–33.

[31] Morita, K., Imai, K. (1997), "A Simple Self-Reproducing Cellular Automaton with Shape-Encoding Mechansim," in *Artificial Life V*, C. G. Langton and K. Shimohara, Eds., Cambridge, MA: MIT Press, pp. 489–496.

[32] Orgel, L.E. (1992), "Molecular Replication," *Nature*, vol. 358, pp. 203–209.

[33] Penrose, L. (1958), "Mechanics of Self-Reproduction," *Ann. Human Genetics*, vol. 23, pp. 59–72.

[34] Perrier, J.-Y., Sipper, M., Zahnd, J. (1996), "Toward a Viable Self-Reproducing Universal Computer," *Physica D*, vol. 97, pp. 335–352.

[35] Pesavento, U. (1995), "An Implementation of von Neumann's Self-Reproducing Machine," *Artificial Life*, vol. 2 no. 4, pp. 337–354.

[36] Ray, T.S. (1992), "Evolution, Ecology and Optimization of Digital Organisms," *Santa Fe Institute Working Paper 92-08-042.*

[37] T. S. Ray (1991), "An Approach to the Synthesis of Life," in [22], pp. 371–408.

[38] Reggia, J. A., Armentrout, S., Chou, H.H., and Peng, Y. (1993), "Simple Systems That Exhibit Self-Directed Replication," *Science*, vol. 259, pp. 1282–1288.

[39] Richards, F.C., Meyer, T.P., and Packard, N.H. (1990), "Extracting Cellular Automaton Rules Directly from Experimental Data," *Physica D*, vol. 45, pp. 189–202.

[40] Sipper, M. (1995), "Studying Artificial Life Using a Simple, General Cellular Model," *Artificial Life*, vol. 2, no. 1, pp. 1–35.

[41] Smith, A.R. (1991), "Simple Nontrivial Self-Reproducing Machines," in [22], pp. 709–725.

[42] Taub, A.H. (1961), *John von Neumann: Collected Works. Volume V: Design of Computer, Theory of Automata and Numerical Analysis*, Oxford: Pergamon Press.

[43] Tempesti, G. (1995), "A New Self-Reproducing Cellular Automaton Capable of Construction and Computation," in *ECAL95: Proceedings of the Third European Conference on Artificial Life,* F. Moran, A. Moreno, J.J. Morelo, and P. Chacon, Eds., Springer, pp. 555–563.

[44] Thatcher, J.W. (1970), "Universality in the von Neumann Cellular Model," in [4], pp. 132–186.

[45] Vitányi, P. M. B. (1973), "Sexually Reproducing Cellular Automata," *Mathematical Biosciences*, vol. 18, pp. 23–54.

[46] von Neumann, J. (1951), "The General and Logical Theory of Automata," in [42], pp. 288–328.

[47] von Neumann, J. (1966), *Theory of Self-Reproducing Automata*, A. Burks, Ed., University of Illinois Press.

[48] Wolfram, S. (1994), *Cellular Automata and Complexity*, Reading, MA: Addison-Wesley.